Renew Suite: A Tool to Compose Individual Renew Tool Sets Based on a Plugin Concept*

Marcel Hansson, Daniel Moldt, and Laif-Oke Clasen

University of Hamburg, Faculty of Mathematics, Informatics and Natural Sciences,
Department of Informatics http://www.paose.de

Keywords: Renew · Modularization · Composition · Plugin Architecture · Petri Nets · Tool Set.

Tools are becoming increasingly complex. One solution for large tools is decomposing it into smaller feature sets or components. The plugin concept is one of the conceptual solutions for customizing individual components [7] (see e.g. plugins for browsers or software development environments). At language level, this is addressed by the concept of modularization. In Java, this is the Java Platform Module System (JPMS) [9], which was introduced with Release 9.

Work on a tool environment for Petri nets has been underway in Hamburg for over 25 years: Renew[1] [6]. The strength of Renew lies in the modeling, simulation and analysis of Petri net models. Other modeling techniques can also be covered. As a growing software system, Renew requires both the individual application-related components, which can be put together individually (as plugins), and modularization using JPMS [3, 5, 8], which makes it possible to establish a clean software architecture.

Due to the numerous variants, there are incompatible components that cannot be used simultaneously. This applies to the commands, the interface arrangements, the formal foundations and the application-related use (in particular redundancies).

To address the static and dynamic composition of our Renew components, we introduce a new concept to Renew: Suite[2] [10] With the help of the Suite concept, the above-mentioned aspects are addressed by selecting individual components and composing them into a concrete tool variant, a suite. The development followed a prototyping approach [11] and was carried out by students as part of normal teaching projects at our department, in particular under the guidance of the first author. In addition to the static composition, dynamic composition is possible, by adding and removing components at runtime. A conceptual basis is the use of the plugin architecture described in [2,4]. However, this Petri net-based model cannot be used directly by users, as complex models would have to be implemented. To simplify matters, the details of the implementation were hidden from users and restricted to purely plugin-based use.

However, the plugin architecture used in Renew still required the use of complex configuration files and their management. Each plugin has its own con-

* Supported by participants of our teaching project classes and many student theses.

[1] www.renew.de

[2] https://en.wikipedia.org/wiki/Software_suite

figuration files with various properties. These properties are often hidden for the usual user despite being potentially useful for their use case. The SUITE concept and the associated plugins provide users with a way to configure these predetermined properties.

The management of dependencies involve a great deal of effort, which ultimately cannot be meaningfully taken over by users. The SUITE concept and the associated plugins can be used to create pre-configurations that replace the previous manually composed and configured RENEW variants for individual usage scenarios. Thanks to the inherent dependency management that is adopted in this way, the composed plugins are compatible and usable. Changes can also be made during runtime using the underlying JPMS, making it easy to switch between different tool configurations. The JPMS allows Java code to be cleanly unloaded at runtime and therefore also plugins.

By means of accompanying descriptions, users get familiar with the services of the respective suites and the configuration options. A graphical user interface is available for experienced users so that they can configure RENEW by creating their own suites.

With regard to the technical basics, it should be noted that the combination of plugins from an application and architecture perspective as well as modules or layers relating to JPMS is of central importance for the implementation. The components of SUITE are three plugins: the underlying framework, the GUI and the creation and management of suites. A headless use is possible, allowing SUITE to configure RENEW for various applications, like container or server applications. RENEW SUITE supports any location of the plugins in the system and can therefore easily integrate plugins from MULAN, a Petri net-based multi-agent architecture [1]. It should be emphasized that non-modularized plugins, such as those still available in MULAN, can also be integrated into a suite. However, the advantages of the JPMS regarding a clean unloading of a plugin, do not apply then.

The possibility of creating plugin configurations that has been implemented so far already makes it possible to significantly streamline the RENEW user interface. Since plugins themselves can be fairly large and can provide several different functionalities at once, the option of deactivating individual functions of plugins via the SUITE concept will be implemented in the further course of development. An initial release of the SUITE concept is planned for an upcoming release of RENEW.

References

1. Cabac, L.: Modeling Agent Interaction Protocols with AUML Diagrams and Petri Nets. Diploma thesis, University of Hamburg, Department of Computer Science, Vogt-Kölln Str. 30, D-22527 Hamburg (Dec 2003)
2. Cabac, L., Duvigneau, M., Moldt, D., Rölke, H.: Multi-agent concepts as basis for dynamic plug-in software architectures. In: Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005). pp. 1157–1158 (2005), https://doi.org/10.1145/1082473.1082671

3. Clasen, L.O., Moldt, D., Hansson, M., Willrodt, S., Voß, L.: Enhancement of Renew to version 4.0 using JPMS. In: Köhler-Bußmeier, M., Moldt, D., Rölke, H. (eds.) Proceedings of the International Workshop on Petri Nets and Software Engineering 2022 co-located with the 43rd International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS 2022), Bergen, Norway, June 20th, 2022. CEUR Workshop Proceedings, vol. 3170, pp. 165–176. CEUR-WS.org (2022), https://ceur-ws.org/Vol-3170
4. Duvigneau, M.: Konzeptionelle Modellierung von Plugin-Systemen mit Petrinetzen, Agent Technology – Theory and Applications, vol. 4. Logos Verlag, Berlin (2010), http://www.logos-verlag.de/cgi-bin/engbuchmid?isbn=2561&lng=eng&id=
5. Janneck, J.R.: Modularizing a Plugin System Using Java Modules: Application to a Medium-Sized Open-Source Project. Master thesis, University of Hamburg, Department of Informatics, Vogt-Kölln Str. 30, D-22527 Hamburg (Mar 2021)
6. Kummer, O., Wienberg, F., Duvigneau, M., Cabac, L., Haustermann, M., Mosteller, D.: Renew – the Reference Net Workshop (Feb 2023), http://www.renew.de/, release 4.1
7. Mayer, J., Melzer, I., Schweiggert, F.: Lightweight plug-in-based application development. In: Aksit, M., Mezini, M., Unland, R. (eds.) Objects, Components, Architectures, Services, and Applications for a Networked World. pp. 87–102. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
8. Moldt, D., Johnsen, J., Streckenbach, R., Clasen, L., Haustermann, M., Heinze, A., Hansson, M., Feldmann, M., Ihlenfeldt, K.: RENEW: modularized architecture and new features. In: Gomes, L., Lorenz, R. (eds.) Application and Theory of Petri Nets and Concurrency - 44th International Conference, PETRI NETS 2023, Lisbon, Portugal, June 25-30, 2023, Proceedings. Lecture Notes in Computer Science, vol. 13929, pp. 217–228. Springer Nature Switzerland AG, Cham, Switzerland (2023), https://doi.org/10.1007/978-3-031-33620-1_12
9. Oracle: Project Jigsaw (9 2017), https://openjdk.java.net/projects/jigsaw/, [Online; accessed 19.05.2022]
10. Rugaber, S.: A tool suite for evolving legacy software. In: 1999 International Conference on Software Maintenance, ICSM 1999, Oxford, England, UK, August 30 - September 3, 1999. pp. 33–39. IEEE Computer Society (1999). https://doi.org/10.1109/ICSM.1999.792496, https://doi.org/10.1109/ICSM.1999.792496
11. Wilde, T., Hess, T.: Forschungsmethoden der wirtschaftsinformatik. Wirtschaftsinf. **49**(4), 280–287 (2007). https://doi.org/10.1007/S11576-007-0064-Z, https://doi.org/10.1007/s11576-007-0064-z